AFRL-IF-RS-TR-2003-194
Final Technical Report
August 2003

A MODEL-BASED REAL-TIME INTRUSION DETECTION SYSTEM FOR LARGE SCALE HETEROGENEOUS NETWORKS

University of California at Santa Barbara

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. F252

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2003-194 has been reviewed and is approved for publication.

APPROVED:

/s/

LUIGI SPAGNUOLO
Project Engineer

FOR THE DIRECTOR: /s/

WARREN H. DEBANY JR., Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	AUGUST 2003	Final Apr 98 – Dec 02

4. TITLE AND SUBTITLE
A MODEL-BASED REAL-TIME INTRUSION DETECTION SYSTEM FOR LARGE SCALE HETEROGENEOUS NETWORKS

5. FUNDING NUMBERS
C - F30602-97-1-0207
PE - 62301E
PR - F252
TA - 40
WU - 21

6. AUTHOR(S)
Richard A. Kemmer and Giovanni Vigna

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of California at Santa Barbara
Department of Computer Science
Santa Barbara California 93106

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFRL/IFGB
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2003-194

11. SUPPLEMENTARY NOTES
AFRL Project Engineer: Luigi Spagnuolo/SNHS/(781) 377-4249/ Luigi.Spagnuolo@hanscom.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***

This final report from the University of California Santa Barbara for their grant entitled “A Model-Based Real-Time Intrusion Detection System for Large Scale Heterogeneous Networks” describes the approach and accomplishments of the research effort. The primary objective for this research was to determine the feasibility of using the State Transition Analysis Technique (STAT) to model and detect Intrusions in large-scale, heterogeneous networks. An additional goal was to develop an extensible framework that supports the development of new STAT-based intrusion detection systems to match new domain and environments. Finally, a goal of this project was to provide a communication and control infrastructure that is able to collect the alerts produced by the sensors, control their configuration, and coordinate their response.

14. SUBJECT TERMS
Real-Time, Large Scale Networks, Intrusion Detection, State Transition Analysis Technique, STAT

15. NUMBER OF PAGES
11

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Table of Contents

Final Report
DARPA grant F30602-97-1-0207
AO Number: F252
February 27, 2003
A Model-based Real-time Intrusion Detection System for Large Scale Heterogeneous Networks

Richard A. Kemmerer, PI
Giovanni Vigna, Co-PI

Department of Computer Science
University of California
Santa Barbara, CA 93106
[kemm,vigna]@cs.ucsb.edu

1 Objective

The primary objective for this research was to demonstrate the feasibility of using the State Transition Analysis Technique (STAT) to model and detect intrusions in large-scale, heterogeneous networks. An additional goal was to develop an extensible framework that supports the development of new STAT-based intrusion detection systems to match new domain and environments. Finally, a goal of this project was to provide a communication and control infrastructure that is able to collect the alerts produced by the sensors, control their configuration, and coordinate their responses.

2 Approach

Intrusion detection systems (IDSs) analyze information about the activities performed in a computer system or network, looking for evidence of malicious behavior. Attacks against a system manifest themselves in terms of events. These events can be of a different nature and level of granularity. For example, they may be represented by network packets, operating system calls, audit records produced by the operating system auditing facilities, or log messages produced by applications. The goal of intrusion detection systems is to analyze one or more event streams and identify manifestations of attacks.

The intrusion detection community has developed a number of different tools that perform intrusion detection in particular domains (e.g., hosts or networks), in specific environments (e.g., Windows NT or Solaris), and at different levels of abstraction. These tools suffer from two main limitations: they are developed *ad hoc* for certain types of domains and/or environments, and they are difficult to configure, extend, and control remotely. In the specific case of signature-based intrusion detection systems (e.g., Snort), the sensors are equipped with a number of signatures that are matched against a stream of incoming events. Most systems are initialized with a set of signatures at startup time. Updating the signature set requires

stopping the sensor, adding new signatures, and then restarting execution. Some of these tools provide a way to enable/disable some of the available signatures, but few systems allow for the dynamic inclusion of new signatures at execution time. In addition, the *ad hoc* nature of existing tools does not allow one to dynamically configure a running sensor so that a new event stream can be used as input for the security analysis.

Another limitation of existing tools is the relatively static configuration of responses. Normally it is possible to choose only from a specific subset of possible responses. In addition, to our knowledge, none of the systems allow one to associate a response with intermediate steps of an attack. This is a severe limitation, especially in the case of distributed attacks carried out over a long time span.

Finally, the configuration of existing tools is mostly performed manually and at a very low level. This task is particularly error-prone, especially if the intrusion detection sensors are deployed across a very heterogeneous environment and with very different configurations.

Our approach to the problem is to develop a framework that defines a domain-independent analysis engine based on a characterization of attacks in terms of states and transitions between states. The framework can be extended in a well-defined way to match new domains, new event sources, and new responses. Intrusion detection applications are built by composing independent building blocks. The resulting set of applications is a software family that shares a common configuration and control framework. This enables the dynamic configuration of a wide range of characteristics.

The framework is centered around the State Transition Analysis Technique. According to this methodology, attack scenarios are represented as a sequence of transitions that characterize the evolution of the security state of a system. In an attack scenario states represent snapshots of a system's security-relevant properties and resources. A description of an attack has an "initial" starting state and at least one "compromised" ending state. States are characterized by means of assertions, which are predicates on some aspects of the security state of the system. For example, in an attack scenario describing an attempt to violate the security of an operating system, assertions would state properties such as file ownership, user identification, or user authorization. Transitions between states are annotated with "signature actions" that represent the key actions that if omitted from the execution of an attack scenario would prevent the attack from completing successfully. For example, in an attack scenario describing a network port scanning attempt, a typical signature action would include the TCP segments used to test the TCP ports of a host. The characterization of attack scenarios in terms of state and transitions allows for an intuitive graphic representation by means of state transition diagrams (STDs).

Attacks modeled using the STAT technique are represented using the STATL language. STATL provides constructs to represent an attack as a composition of states and transitions. Because it is possible for several occurrences of the same attack to be active at the same time, a STATL attack scenario has operational semantics in terms of a set of "instances" of the same scenario "prototype". The scenario prototype represents the scenario's definition and global environment, and a scenario instance represents a particular attack that is currently in progress.

The evolution of the set of instances of a scenario is determined by the type of transitions in the scenario definition. A transition can be consuming, nonconsuming, or unwinding. A nonconsuming transition is used to represent a step of an occurring attack that does not prevent further occurrences of attacks from spawning from the transition's source state. Therefore, when a nonconsuming transition fires, the source state remains valid, and the destination state becomes valid too. For example, if an attack has two steps that are the creation of a symbolic link to a SUID program and the execution of the program through the created link, then the second step does not invalidate the previous state. That is, another execution of the program through the same link may occur. Semantically, the firing of a nonconsuming transition causes the creation of a new scenario instance. The original instance is still in the original state, while the new instance is in the state that is the destination state of the fired transition. In contrast, the firing of a consuming transition makes the source state of a particular attack occurrence invalid. Semantically, the firing of a consuming transition does not generate a new scenario instance; it simply changes the state of the original one. Unwinding transitions represent a form of "rollback" and they are used to describe events and

conditions that may invalidate the progress of one or more scenario instances and require the return to an earlier state. For example, the deletion of a file may invalidate a condition needed for an attack to complete, and, therefore, a corresponding scenario instance may be brought back to a previous state, such as before the file was created.

STATL attack descriptions are executed by the language runtime module, called the STAT Core. The Core implements the concepts of state, transition, instance, timer, etc. In addition, the STAT Core is responsible for obtaining events from the target environment, and matching this event stream against the actions and assertions corresponding to transitions in the active attack scenarios.

The STATL language and the Core runtime are domain-independent. They do not support any domain-specific features that may be necessary to perform intrusion detection analysis in particular domains or environments. For example, network events such as an IP packet or the opening of a TCP connection are not represented in STATL natively. Instead, the STAT Framework provides a number of mechanisms to extend the STATL language and the runtime to match the characteristics of a specific target domain.

The first step in the extension process is to create the events and types that characterize a target domain. A STAT event is the representation of an element of an event stream to be analyzed. For example, an "IP" event may be used to represent an IP datagram that has been sent on a link. The event stream is composed of IP datagrams and other event types, such as Ethernet frames and TCP segments. Basic event types can be composed into complex tree structures. For example, it is possible to express encapsulation (e.g., Ethernet frames that encapsulate IP datagrams, which, in turn, contain TCP segments) using a tree of events.

A set of events and types that characterize the entities of a particular domain is called a "Language Extension". The name comes from the fact that the events and types defined in a Language Extension can be used when writing a STATL scenario once they are imported using the "use" STATL keyword. For example, if the "IP" event and the "IPAddress" type are contained in a Language Extension called "tcpip". Then, by using the expression "use tcpip" it is possible to use IP events and IPAddress objects in STATL attack scenario descriptions.

The events and types defined in a Language Extension must be made available to the runtime. Therefore, Language Extensions are compiled into dynamically linked libraries (i.e., a ".so" file in a UNIX system or a DLL file in a Windows system). The Language Extension libraries are then loaded into the runtime whenever they are needed by a scenario.

Attack scenarios are written in STATL, extended with the relevant Language Extensions. For example, a signature for a port scanning attack can be expressed in STATL extended with the "tcpip" Language Extension. Then, STATL attack scenarios are automatically translated into C++ and compiled into dynamically linked libraries, called "Scenario Plugins". When loaded into the runtime, Scenario Plugins analyze the incoming event stream looking for events or sequences of events that match the attack description.

Once Language Extensions and Scenario Plugins are loaded into the Core it is necessary to start collecting events from the environment and passing them to the STAT Core for processing. The input event stream is provided by one or more "Event Providers". An Event Provider collects events from the external environment (e.g., by obtaining packets from the network driver), creates STAT events as defined in one or more Language Extensions, and inserts these events into the event queue of the STAT Core.

A STAT Core runtime equipped with Language Extensions, Scenario Plugins, and Event Providers represents a functional intrusion detection system. However, the STAT Framework also provides support for the definition of "Response Modules". A Response Module is a library of actions that may be associated with the evolution of a scenario. For example, a network-based response action could reset a TCP connection, or it could send an email to the Network Security Officer. Response Modules are compiled into dynamically linked libraries that can be loaded into the runtime at any moment. Functions defined in a Response Module can be associated with any of the states defined in a Scenario Plugin that has been loaded in the runtime. This mechanism provides the ability to associate different types of response functions with the intermediate steps of an intrusion.

STAT-based sensors can operate as stand-alone IDSs or they can be integrated in a control and communication infrastructure, called MetaSTAT. The MetaSTAT infrastructure has a number of different com-

ponents. First of all, the CommSTAT library allows components in the infrastructure to exchange alert messages and control directives in a secure way. CommSTAT messages are exchanged of SSL-protected TCP connections and follow the standard Intrusion Detection Message Exchange Format (IDMEF). The original IDMEF definition has been extended to include STAT-related control messages that are used to control and update the configuration of STAT-sensors. For example, messages to transfer STAT modules (e.g., Language Extensions) across the infrastructure, as well as messages to load modules into sensors have been defined.

STAT-based sensors participation in the MetaSTAT infrastructure is mediated by a component, called the STAT Proxy, that is responsible for managing a set of sensors for a specific host. The STAT Proxy is responsible for the installation and uninstallation of modules, and for the routing of messages and control directives to and from the connected sensors. In addition, the STAT Proxy supports the integration of third-party tools that are not based on the STAT Framework.

STAT Proxies are connected to one or more Controllers, which are management applications. A Controller is used by the MetaSTAT administrator to perform activation and reconfiguration of STAT-sensors, and to collect sensor alerts.

The alerts collected by a Controller are passed to a Collector component that is responsible for storing the alerts in a persistent way. The IDMEF alerts are stored in a MySQL relational database. The content of the database can be displayed and queried using the Alert Viewer, which provides a user-friendly graphic interface.

Controllers can be composed hierarchically to achieve cross-domain control and monitoring.

In summary, the STAT Framework supports the development of intrusion detection systems for different, heterogeneous domains and environment and the MetaSTAT infrastructure provides the means to control the sensors deployed in large-scale networks, collecting the results of their analysis in a central repository.

3 Accomplishments

The STAT Framework has been completed. In particular, the STATL language has been designed and a STATL-to-C++ translator has been developed. In addition, the STAT Core has been implemented. The STAT Core fully support the dynamic loading of modules and the run-time reconfiguration of sensors.

A number of sensors have been developed using the framework.

- **NetSTAT** is a network-based intrusion detection system, that uses the traffic sniffed on a local network as input.
- **USTAT** is a host-based intrusion detection system, that uses the audit records produced by Sun Microsystems' Solaris Basic Security Module (BSM) Packages as input.
- **LinSTAT** is a host-based intrusion detection system, that uses an event stream provided by a modified version of the SNARE kernel auditing module as input.
- **WinSTAT** is a host-based intrusion detection system, that uses Windows NT event logs as input.
- **logSTAT** is a host-based intrusion detection system, that uses UNIX syslog events as input.
- **webSTAT** is a host-based intrusion detection system, that uses web server application logs as input.
- **AlertSTAT** is an intrusion detection system that uses IDMEF alerts as input and performs alert fusion, aggregation, and correlation.

The development of each sensor required the design and implementation of one or more Language Extensions to represent the event stream being analyzed, the development of one or more Event Providers that collect the events from the operational environment, and a number of Scenario Plugins that model different

attacks. The STATL language and the Core runtime are shared by all the sensors, resulting in a simplified design, reduced development time, and code reuse.

The byproduct of the development of these sensors is a database of attacks that include vulnerable software, exploits, attack traces, and detection signatures. The software for the STAT Framework and all the tools is available at the project web site (http://www.cs.ucsb.edu/~rsg/STAT).

The MetaSTAT infrastructure has been completed. The CommSTAT communication library has been developed in both C and Java to allow for the integration of application developed in different languages. The Controller has been developed in a text based form and in a Web-based graphic version. The Collector has also been developed and a schema for efficient storing of IDMEF alerts has been developed. The Collector uses a generalized DTD-to-SQL schema translation process that allows for the automatic generation of the SQL schema from the IDMEF DTD. This feature is important because the IDMEF format is currently going through the standardization process and a generative approach accommodates for changes in the format without impacting the interface of the analysis tools. The Alert Viewer has also been developed and optimized for performance. The software for the MetaSTAT infrastructure is available at the project web site (http://www.cs.ucsb.edu/~rsg/STAT).

3.1 Papers Published

This research produced a number of publications in International Journals, Conference, and Workshops. A list of the publications related to this project follows.

R.A. Kemmerer, "NSTAT: A Model-based Real-time Network Intrusion Detection System," Computer Science Dep., University of California Santa Barbara, Technical Report TRCS97-18, November 1997.

G. Vigna and R.A. Kemmerer, "NetSTAT: A Network-based Intrusion Detection Approach," in Proceedings of the 14th Annual Computer Security Application Conference, Scottsdale, Arizona, December 1998 (This paper won the Outstanding Paper Award.)

G. Vigna and R.A. Kemmerer, "NetSTAT: A Network-based Intrusion Detection System," Journal of Computer Security, 7(1), IOS Press, 1999.

G. Vigna, S.T. Eckmann, and R.A. Kemmerer, "The STAT Tool Suite," in Proceedings of DISCEX 2000, Hilton Head, South Carolina, January 2000, IEEE Press.

G. Vigna, S.T. Eckmann, and R.A. Kemmerer, "Attack Languages," in Proceedings of the IEEE Information Survivability Workshop, Boston, MA, October 2000.

S.T. Eckmann, G. Vigna, and R.A. Kemmerer, "STATL: An Attack Language for State-based Intrusion Detection," in Proceedings of the ACM Workshop on Intrusion Detection, Athens, Greece, November 2000.

S.T. Eckmann, G. Vigna, and R.A. Kemmerer, "STATL Syntax and Semantics," Computer Science Dep., University of California Santa Barbara, Technical Report TRCS20-19, December 2000.

G. Vigna, R.A. Kemmerer, and P. Blix, "Designing a Web of Highly-Configurable Intrusion Detection Sensors," in Proceedings of the Workshop on Recent Advances in Intrusion Detection (RAID 2001), Davis, CA, October 2001.

S.T. Eckmann, "Translating Snort rules to STATL scenarios", presented at the 4th International Symposium on Recent Advances in Intrusion Detection (RAID 2001), Davis, CA, October 2001, LNCS 2212, pp. 69-84.

S.T. Eckmann, G. Vigna, and R.A. Kemmerer, "STATL: An Attack Language for State-based Intrusion Detection," Journal of Computer Security, vol. 10, no. 1/2, pp. 71-104, 2002.

R.A. Kemmerer and G. Vigna, "Intrusion Detection", in IEEE Computer, Special Issue on Security and Privacy, April 2002.

C. Kruegel, F. Valeur, G. Vigna, and R.A. Kemmerer, "Stateful Intrusion Detection for High-Speed Networks," IEEE Symposium on Security and Privacy, Oakland, CA, May 2002.

G. Vigna, B. Cassell, and D. Fayram, "An Intrusion Detection System for Aglets", in Proceedings of the International Conference on Mobile Agents (MA '02), Barcelona, Spain, October 2002.

G. Vigna, F. Valeur, J. Zhou, and R.A. Kemmerer, "Composable Tools For Network Discovery and Security Analysis," Proceedings of the Eighteenth Annual Computer Security Applications Conference (ACSAC'02), Las Vegas, Nevada, pp. 14-24, December 2002.

3.2 Students Graduated

The following PhD students were supported by this research:

- Paul Kolano 1999, currently at NASA/Ames
- Zhe Dang 2000, currently at Washington State University
- Andre dos Santos 2000, currently at Georgia Tech
- Steven Eckmann 2002, independent consultant

Numerous Masters degree students were also supported by this research.

4 Technology Transition

The STAT toolset has been used in a number of technology transfer efforts. Participating in these efforts provided feedback on the effectiveness of the STAT approach. In addition, limitations that were discovered when the systems were deployed in a real-life setting were addressed to improve the overall performance of the system.

A first form technology transfer is the participation to a number of DARPA-sponsored intrusion detection evaluation efforts.

In 1998 we participated in the off-line intrusion detection evaluation organized by Lincoln Labs/MIT. NetSTAT and USTAT were run against network dumps and BSM files, respectively. The results were then delivered to LL for evaluation. The STAT tools performed at the highest level in the evaluation. The lessons learned by participating in this effort were the motivation factor for re-designing the STAT sensors as extensions of a common framework.

The attack database built for testing NetSTAT and USTAT was given to the Air Force Research Laboratory (AFRL) for use in their 1998 real-time evaluation. In addition to taking part in the evaluation, we gave support in fixing problem with the AFRL testbed network setup.

The code for the USTAT preprocessor was also delivered to the "Fraud and Intrusion Detection for Financial Information Systems using Meta-Learning Agents" project at Columbia University. They used the preprocessor as part of their preparation for the DARPA off-line evaluation.

In 1999, USTAT, NetSTAT, and WinSTAT were used to participate in the 1999 LL/MIT IDS evaluation. These tools were a complete re-implementation of the original tools based on the STAT Framework. Again, the STAT toolset performed at the highest levels.

As a byproduct of our participation in the effort, we helped the MIT/Lincoln Laboratory group with the debugging of their training data.

USTAT and NetSTAT were also delivered to the AFRL for use in their 1999 real-time evaluations.

Transfer of ideas and technologies were also performed by participating in the DARPA Attack Language subgroup. A paper on STATL was distributed to the subgroup and a paper on attack languages was published.

Throughout the project's lifetime we collaborated with SRI's Emerald (project "Analysis and Response for Intrusion Detection in Large Networks") to determine how to approach the integration of USTAT, NSTAT, and NetSTAT in SRI's EMERALD environment to demonstrate interoperability.

The code for USTAT, NetSTAT, and WinSTAT was delivered to the TIC for installation in their testbed network. In addition, the code was delivered to Dan Schnackenberg at Boeing to be integrated into the JICPAC exercises.

In 2002, we participated in the validation experiment of the CyberPanel program. We provided the USTAT and WinSTAT sensors for inclusion in the testbed network and we closely collaborated with the exercise organizers to match their requirements in terms of tool configuration, scenarios, and type and format of response.

In addition, we developed the AlertSTAT correlator to demonstrate the applicability of the STAT analysis technique to alert fusion and correlation. Even though the tool was developed in a very short time frame, AlertSTAT outperformed (in terms of the effectiveness/accuracy ratio) correlator tools that were developed in longer time and with greater development effort.

The lessons learned by participating in the correlation effort were also applied to the analysis of the data collected by the AFRL's AFED tool. A new tool, called AFEDSTAT, was developed to match the AFED event model and to fulfill AFRL's analysis requirements.

Finally, as a form of technology transition, the code in source and binary form is available at the project's web site (http://www.cs.ucsb.edu/~rsg/STAT), together with detailed documentation.